



Controle de Estados na Web




lá, tudo bem?

Veremos nesta aula alguns recursos importantes que podem ser utilizados para tratar estados de jogos digitais na web. Podemos receber informações do navegar através da URL e utilizá-las em nossos programas, bem como armazenar e tratar sessões no navegador do usuário por meio do JavaScript.

Neste resumo, veremos a sintaxe de aplicação e exemplos rápidos utilizando QueryString, a classe URLSearchParams e o objeto sessionStorage.

Utilizando QueryString

Uma QueryString é uma cadeia de caracteres contendo um conjunto de informações contidas na URL utilizadas para recuperar informações no servidor web. Utilizar a QueryString é uma forma clássica de controlar o estado da página. Elas são nada mais do que conjuntos de chaves/valores anexados na URL.

Após a URL de uma aplicação, usa-se o caractere **?** para indicar o início da QueryString. Em seguida adiciona-se a sequência de valores usando a sintaxe **Chave=Valor**. Para passarmos mais de um conjunto, os mesmos devem ser concatenados usando o caractere **&**.

Por exemplo, se estivéssemos buscando por picaretas de ferro no site da loja de um jogo de construção e sobrevivência, a QueryString seria algo como:

https://light-craft.glitch.me/?**item**=picareta&**material**=ferro



Neste exemplo estamos passando dois parâmetros junto com a URL: **item** e **material**. Essas informações podem ser tratadas no servidor web ou na aplicação cliente com JavaScript.

Um programa JavaScript para acessar esta QueryString da barra de endereços do navegador pode ser feito utilizando a propriedade **search** do objeto pré-definido **location**. Tal propriedade armazena a cadeia de caracteres da QueryString.

Como o valor contido em search é uma cadeia de caracteres, precisamos separar os parâmetros e seus valores. Para isso utilizamos a classe **URLSearchParams**.

URLSearchParams define vários métodos de utilização para trabalhar com QueryString de uma URL. No trecho de código a seguir, veremos como acessar a informação com o método URLSearchParams.**get**():



```
// Recuperar a QueryString passada ao navegador
// Ela está armazenada na propriedade search do objeto location
let queryString = location.search;

// Instanciando a classe URLSearchParams
// para que possamos acessar os valores dos parâmetros
let paramsBusca = new URLSearchParams(queryString);

// Armazenar os valores dos parâmetros recuperados
// com o método get() de URLSearchParams
let item = paramsBusca.get("item");
let material = paramsBusca.get("material");

// Escrever no console sobre a busca
console.log("Buscando por " + item + " de " + material);
```

Utilizar esse recurso é útil para que seja possível voltar a um estado específico em nosso jogo quando nosso usuário, por algum motivo, atualizar a página no navegador, por exemplo. Além disso, é o formato mais utilizado em buscas na internet e para trocar informações entre sites diferentes.

Controlando Sessões com sessionStorage

Valores enviados nas requisições HTTP com QueryString podem ser modificados pelo usuário e dependendo do nosso projeto, não queremos que isso seja possível. Sendo assim, outra possibilidade de controlar estados do nosso jogo é salvando dados no próprio navegador do usuário.

Isso é possível por meio dos objetos **localStorage** e **SessionStorage,** que permitem salvar pares de chave / valor em um navegador. Nesta unidade aprenderemos a utilizar o SessionStorage.

O objeto Session Storage armazena dados em apenas uma sessão no navegador. A sessão irá durar enquanto o navegador estiver aberto e se mantém ao atualizar a página. No entanto, abrir a mesma página em uma nova aba ou janela irá gerar uma nova sessão. Além disso, os dados são excluídos quando a guia do navegador é fechada.

A sintaxe básica para salvar e recuperar dados em sessionStorage é a seguinte:

```
// Salva os dados em sessionStorage
sessionStorage.setItem("chave", valor);

// Obtém os dados de sessionStorage
let data = sessionStorage.getItem("chave");
```

Assim, **chave** é o nome que damos para a informação que será gravada e **valor** será o conteúdo a ser armazenado. Note que a chave deve ser escrita entre aspas.

Por exemplo, se quiséssemos armazenar na sessão do navegador qual o Pokémon atual que o jogador está utilizando em batalha, faríamos:

```
// Salva o Pokémon atual em sessionStorage
```

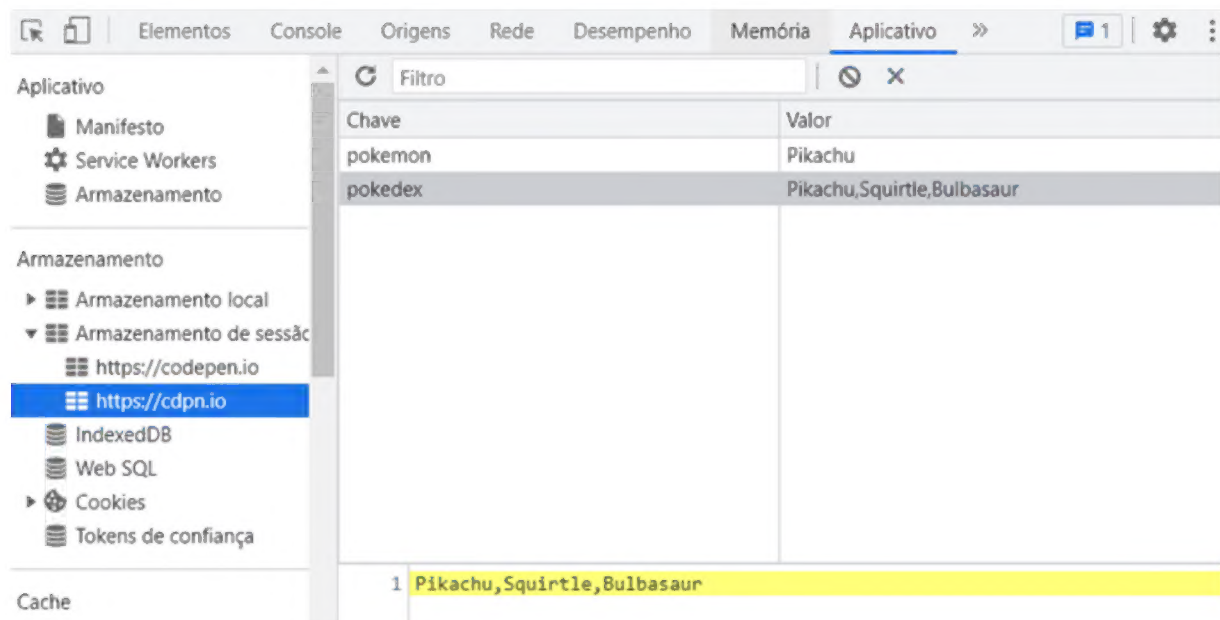
```
sessionStorage.setItem("pokemonAtual", "Pikachu");
```

Isso irá salvar a *string* “Pikachu” em sessionStorage com a chave identificadora de pokemonAtual. Então, se quiséssemos acessar os valores salvos na sessão em nosso programa, basta usar o método getItem():

```
// Obtém o pokémon de sessionStorage
let pokemonAtual = sessionStorage.getItem("pokemonAtual");
```



Veja na figura a seguir um exemplo de como os valores e chaves ficam armazenados no navegador Chrome, nas ferramentas do desenvolvedor.



Um ponto que devemos observar é que os dados salvos em sessionStorage serão sempre armazenados como do tipo *string*. Com isso, sempre que recuperamos um dado dela, devemos fazer o tratamento e conversões de tipos necessárias.

Outro ponto é que a informação fica **gravada e acessível** somente para o domínio atual da aplicação. Ou seja, sites de diferentes domínios não compartilham o que está gravado na sessão do navegador.

Finalizamos mais um resumo Padawan! Agora você já sabe como fazer para que os jogadores de seus jogos não fiquem tristes se atualizarem o navegador, perdendo informações sobre seu progresso no jogo. O que você viu aqui é muito útil, pois trata-se também de uma possibilidade para que

nossos jogos possam continuar funcionando de forma off-line, caso o jogador fique sem conexão de internet em alguns momentos de gameplay.

Forte abraço!

Atividade Extra

Leia o artigo abaixo para consolidar seus conhecimentos sobre o SessionStorage do HTML5.

- **MDN Web Docs JavaScript Events Reference**, disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/API/Window/sessionStorage>.

Referência Bibliográfica

LAWSON B. & SHARP R. **Introdução ao Html 5.** Alta Books: Rio de Janeiro. 2011.

Ir para exercício